Theses and Dissertations   1. Thesis and Dissertation Collection, all items

1972-12

Conceptual dependency structure in the NLP natural language processor.

Hull, Bradley Wayne

http://hdl.handle.net/10945/16046

CONCEPTUAL DEPENDENCY STRUCTURES
IN THE NLP
NATURAL LANGUAGE PROCESSOR

Bradley Wayne Hull

NAVAL POSTGRADUATE SCHOOL
Monterey, California
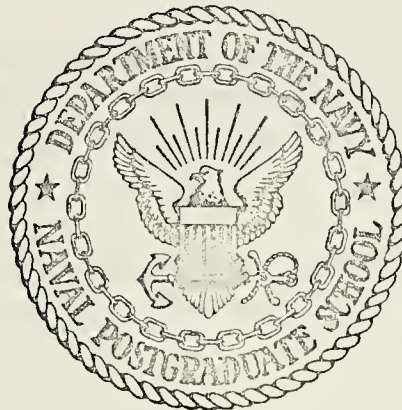
THESIS

CONCEPTUAL DEPENDENCY STRUCTURES
IN THE NLP
NATURAL LANGUAGE PROCESSOR

by

Bradley Wayne Hull

Thesis Advisor: G. E. Heidorn

December 1972

T15 948

Conceptual Dependency Structures
in the NLP
Natural Language Processor

by

Bradley Wayne Hull
Lieutenant, United States Navy
B.S., University of Utah, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1972

ABSTRACT

There have been many systems developed for computer process-
ing of natural languages such as English. One of these, known as NLP,
is being developed at the Naval Postgraduate School. Another system,
based on Conceptual Dependency theory, is being developed at Stanford
University. The two systems, while having somewhat similar goals,
use different internal representations of information.

The purpose of this thesis was to devise a means for representing
the structures of Conceptual Dependency theory in NLP, and to develop
methods for conversion of information between NLP's existing repre-
sentation and that of Conceptual Dependency theory.

TABLE OF CONTENTS

I. INTRODUCTION

There has been much research devoted to the development of language processors that will allow users to communicate with computers in a natural language such as English. A number of efforts in this area are reported in Refs. 5, 6, and 7.

Many current researchers feel that the key to developing a useful natural language processor lies in having a "cognitive" theory of language, one which attempts to model the language behavior of human beings. One such theory is that of stratificational linguistics developed by Lamb [2].

This report deals with two natural language processing systems. Although both are being developed generally within the framework of stratificational linguistics, they use different representations for information found in natural language text. The first of these, Natural Language Processor (NLP), is being developed at the Naval Postgraduate School by G. E. Heidorn. It is intended to be a general natural language processing system, and it uses an entity-attribute-value form of information-structure. A specific application--that of producing a GPSS program from an English description of a queuing problem--is being utilized as a vehicle for its development. The other system, being developed at Stanford University by R. C. Schank, is called Conceptual Dependency theory, and defines a structure using

"dependency relations" between concepts to represent the meanings of natural language text.

The objective of the research being reported on in this thesis was to develop, within the framework of NLP, a representation for the dependency structures of Conceptual Dependency theory and also to develop procedures for converting information between that representation and the existing NLP representation.

The remainder of the thesis is divided into four sections. Section II discusses Conceptual Dependency Theory, and Section III discusses NLP. Section IV describes the structure and rules developed in this research. Finally, Section V presents conclusions and recommendations.

II. CONCEPTUAL DEPENDENCY THEORY

This section provides an introduction to Conceptual Dependency
Theory. It should be noted that, since the development of this theory
is an on-going project, the concepts and rules presented in this section
are subject to modification as Schank's work progresses. The material
for this section was taken primarily from Refs. 3 and 4. Some por-
tions, notably definitions, were taken directly from the reference
material.

A.   GENERAL DISCUSSION

Most parsers used in "conversation" machines have been syn-
tactic parsers. That is, they analyze an input sentence and construct
a syntactic (grammatical) network from it. But consider the sentence,
"I hit the boy with the girl with long hair with a hammer with venge-
ance." Clearly the syntactic structure of this sentence will not provide
the information necessary to get at the meaning of it. For example, if
we need to know that it was the hammer that hit the boy, we would have
to use methods more sophisticated than syntactic analysis.

Computer programs that process natural language do so for some
purpose. They need to make use of the information provided by a sen-
tence so that they can respond properly. But whatever the purpose, it
is the meaning of the input sentence that is needed, not its syntactic
structure. A major goal of this work is the attempt to analyze natural

language into meaning (conceptual) structures that are unambiguous representations of the meaning of an input utterance. That is, any two utterances that can be said to mean the same thing, whether they are in the same or different languages, should be characterized in only one way by the conceptual structures. The representation of the content of an utterance then, must be in interlingual terms that are as neutral as possible. The conceptual base is responsible for formally representing the concepts underlying an utterance without respect to the language in which that utterance was encoded. The concept(s) that a given word may have must be found and related in some way to those concepts denoted by other words in a given utterance.

There are two distinct levels of analysis here that are part of a stratified (many-leveled) system. On the <u>sentential</u> level, the utterances of a given language are encoded within the syntactic structure of that language. The basic construction of the sentential level is the sentence. The next higher level in the system is the <u>conceptual</u> level, and the basic construction of this level is the <u>conceptualization</u>. A conceptualization consists of <u>concepts</u> and certain relations that exist between these concepts. Underlying every sentence in a language there exists at least one conceptualization.

B.   CONCEPTS

The basic unit of the conceptualization is the concept. There are three elemental kinds of concepts--nominals, actions, and modifiers.

1. Nominals

Nominals are those things that can be thought of by them-
selves without the need for relating them to other concepts. A word
that is a realization of a nominal concept tends to produce a picture of
that real-world item in the mind of the hearer, so nominal concepts are
called PP's (for picture producer). A PP then, is the concept of a gen-
eral thing, e.g., a man, a book, a mountain; or of a specific thing--
John, New York, or the Grand Canyon.

2. Actions

An action is that which a nominal can be said to be doing. In
order for a concept to qualify as an action (ACT) it must be something
that an animate nominal can do to some object. Thus, since "John hit
Mary" expresses an action that happened to Mary, "hit" is an ACT. In
"John likes Mary" however, nothing happens to Mary, so "like" is not
an ACT.

3. Modifiers

A modifier is a concept that makes no sense without the
nominal or action to which it relates, and serves to specify an attribute
of that nominal or action. Modifiers of nominals are PA's (picture
aiders), and modifiers of actions are AA's (action aiders).

C. DEPENDENCIES

Each of these conceptual categories (PP, ACT, PA, and AA) can
be related in specified ways to one another. These relations are called
dependencies. The rule of thumb in establishing dependency relations

9

between two concepts is whether one item can be understood without the other. A governor can be understood by itself. However, for a conceptualization to exist, even a governor must be dependent on some other concept in that conceptualization.

PP's and ACT's are inherently governing categories, while PA's and AA's are inherently dependents. However, governors can also be dependent, and in order for a conceptualization to exist, this must be the case for at least two governors.

The conceptual base is represented by a linked network of concepts and dependencies that is called a conceptual dependency network. As an example of what such a network looks like, consider the sentence: "John hit his little dog." "John" is the name of an object, so it represents a concept which can be understood by itself, and it is thus a PP. "Hit" represents a concept of action. Each of these concepts is necessary to the conceptualization. Thus, a two-way dependency exists between them. That is, they each act as governors which can be understood by themselves but which must both be present in order to form a conceptualization. The two-way dependency is denoted by $\Longleftrightarrow$. The words "his" and "little" both represent dependent concepts in that in order to understand them it is necessary to hold them in waiting until what they modify appears. "Dog" is the name of a concept which is a PP and is therefore a governor. The PP "dog" is conceptually related to the ACT "hit" insofar as it cannot be understood with respect to the conceptualization except in terms of "hit." This objective dependency

10

is denoted $\overset{\text{o}}{\longleftarrow}$. So the network to this point is:

John $\Longleftrightarrow$ hit $\overset{\text{o}}{\longleftarrow}$ dog

Now the dependents that were waiting for "dog" as governor can be added. "Little" represents a PA that is dependent on "dog." This is called <u>attributive</u> <u>dependency</u> and is denoted by $\uparrow$. The concept given by "his" would appear to be dependent on "dog" as well, and it is, but it is not a simple concept. "His" is really another syntactic represen- tation of the PP "John" that is being used in the syntactic form that indicates possession. The true relation, then, is one PP acting as a dependent modifier to another PP. <u>Prepositional dependency</u> between two PP's is denoted $\Uparrow$ with a label indicating the type of prepositional dependency. (POSS indicates that the governor possesses the depen- dent.) The final network is:

John $\Longleftrightarrow$ hit $\overset{\text{o}}{\longleftarrow}$ dog

$\uparrow$ $\Uparrow$ POSS

little John

D. CONCEPTUAL RULES

The conceptual level is intended to represent the concepts and relations between concepts that underlie natural language utterances. There are formally defined dependency relations between given cate- gories of concepts, and these relations are the <u>conceptual</u> <u>rules.</u> These rules, and only these, make up the formal organization of the conceptual networks of the conceptual level. All of these rules are summarized in Fig. 1 at the end of the section.

1. Actor-ACT Dependency

Since a conceptualization expresses an event, the heart of any conceptualization is the relationship between the actor and the action in that event. Thus rule 1 states that PP's can ACT and when they do there is a mutual dependence between them. For example, in the sentence "John hit Mary," the two-way dependency is shown as

John $\Longleftrightarrow$ hit

2. , Attributive Predication

It is possible to predicate an attribute about a particular PP. This is called an attributive conceptualization and the relationship is denoted by $\Longleftrightarrow$. In order for these items to exist as a conceptualization, each is equally necessary, so the dependency is two-way. For example,

John $\Longleftrightarrow$ tall

3. ACT Membership

It is also possible to predicate ACT membership between two PP's:

John $\Longleftrightarrow$ doctor

4. Attributive Dependency

It is possible to refer to a concept and an attribute of that concept that has already been predicated. In discourse, conceptual attributes are predicated, either explicitly or implicitly, before they are used, to differentiate concepts of the same linguistic name. For example, "the tall man" would only be used to differentiate two men

12

whose relative height is either visually apparent or has been previously

remarked upon in a predication.   "The tall man" would be shown as

$$man \longleftarrow tall$$

5.   Location, Possession, and Containment

Two conceptual objects in the world can be related to each

other in various fashions.   The three principle ones are location, pos-

session, and containment, and these are marked on the ⇑ arrows.

For example:

man	book	water
⇑ LOC	⇑ POSS	⇑ CONT
New York	John	bucket

6.   Objective Dependency

Rule 6 indicates objective dependency.   The PP is related as

object to the ACT which governs it.   For example, the sentence "John

hit Mary" gives the network

$$John \underset{}{\overset{p}{\Longleftrightarrow}} hit \overset{o}{\longleftarrow} Mary$$

(The "p" is placed over the arrow ⟺ to indicate that the event took

place in the "past.")

7.   Recipient Case

The recipient case, which is dependent on the ACT through

the object, is used to denote the transition in possession of the object

from the originator to the recipient.   A concept like "give" or "take"

obviously requires a recipient case, although the original possessor

or the recipient may be only implied or may be unknown.   For example,

in the sentence "I gave the man a book" both the originator and the recipient are known:

$$I \xleftrightarrow{p} \text{give} \xleftarrow{o} \text{book} \quad \xleftarrow{R} \begin{array}{c} \xrightarrow{\text{to}} \text{man} \\ \xleftarrow{} \text{I} \\ \text{from} \end{array}$$
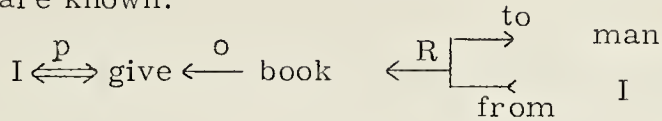
However, in the sentence "the man took a book", the donor is unknown:

$$\text{man} \xleftrightarrow{p} \text{take} \xleftarrow{o} \text{book} \quad \xleftarrow{R} \begin{array}{c} \xrightarrow{} \text{man} \\ \xleftarrow{} (x) \end{array}$$
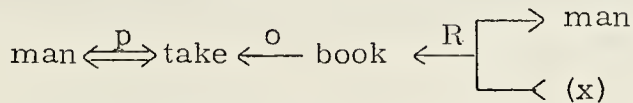
It can be seen that these two conceptualizations look very much alike. Conceptually, the same underlying action has occurred-- the transfer of an object from one person to another. The difference lies only in the direction of transfer. But by realizing that direction can be determined by comparing the actor with the donor (or recipient) of the conceptualization, it would seem reasonable to be able to replace "give" and "take" by a new ACT "TRANS". For example, the first conceptualization above would be realized as

$$I \xleftrightarrow{p} \text{TRANS} \xleftarrow{o} \text{BOOK} \quad \xleftarrow{R} \begin{array}{c} \xrightarrow{\text{to}} \text{MAN} \\ \xleftarrow{} \text{I} \\ \text{from} \end{array}$$
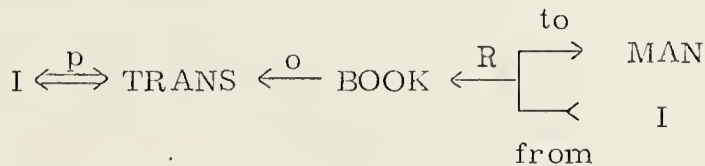
and the second example as

$$\text{MAN} \xleftrightarrow{p} \text{TRANS} \xleftarrow{o} \text{BOOK} \quad \xleftarrow{R} \begin{array}{c} \xrightarrow{\text{to}} \text{MAN} \\ \xleftarrow{} \text{SOMEONE} \\ \text{from} \end{array}$$
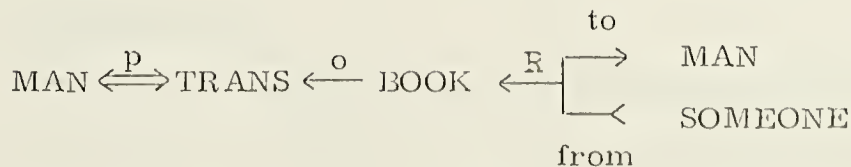
"Give" can then be defined as "TRANS" where actor and donor are the same, and "take" is "TRANS" where actor and recipient are the same.

14

Many other verbs besides "give" and "take" can be realized as "TRANS" plus other requirements (e.g., "buy", "sell", and "steal"), but "give" and "take" are the only ones being presently considered. It would seem that being able to map many concepts into a single concept without losing any information would greatly simplify the conceptual network.

8. Instrumental Case

Rule 8 shows the instrumental case. The instrument of an ACT is a conceptualization, and represents the means by which the main action was completed. Every ACT requires an instrument, but more often than not the instrument is implied rather than explicitly stated. For example, in the sentence "John hit Mary", the instrument is not stated. But in the sentence "John hit Mary by throwing a stick", the entire instrumental conceptualization is present. The conceptual network for this sentence would be

$$
\text{John} \xLeftrightarrow{p} \text{hit} \xleftarrow{\ o\ } \text{Mary} \xleftarrow{\ I\ } \begin{array}{c} \text{John} \\ \Updownarrow \ p \\ \text{throw} \\ \uparrow \ \ o \\ \text{stick} \end{array}
$$
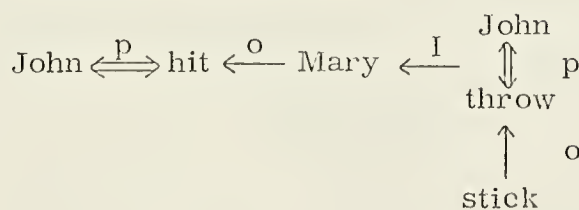
9. Directive Case

The directive case indicates that PP's may serve as the directional indicators of a directional action. In the example above, the action "throw" takes the directive case even though there is no explicit mention of direction. In order for the sentence to completely express the conceptualization, it would have to be "John hit Mary by

throwing a stick at her", and the network would be

$$\text{John} \overset{p}{\Longleftrightarrow} \text{hit} \overset{o}{\longleftarrow} \text{Mary} \overset{I}{\longleftarrow} \begin{array}{c} \text{John} \\ \Updownarrow p \\ \text{throw} \end{array}$$

$$\begin{array}{c} \uparrow o \\ \text{stick} \overset{D}{\longleftarrow} \end{array} \begin{array}{c} \longrightarrow \text{Mary} \\ \longleftarrow \text{John} \end{array}$$
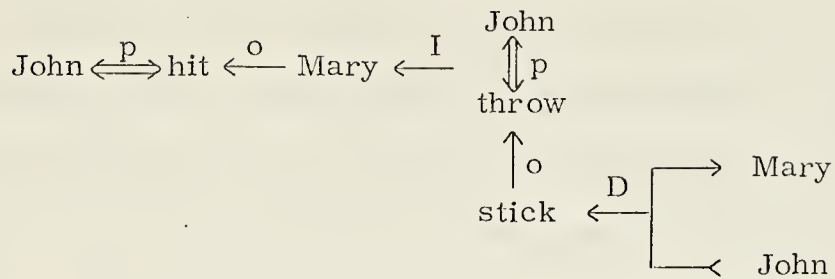
10. Causality

Causality is denoted $\Uparrow$, and indicates that the governing conceptualization caused the dependent conceptualization to happen. As an example, the sentence "John was sad because Mary left him" yields the network

$$\begin{array}{c} \text{Mary} \overset{p}{\Longleftrightarrow} \text{go} \overset{D}{\longleftarrow} \\ \Uparrow \\ \text{John} \overset{p}{\Longleftrightarrow} \text{sad} \end{array} \begin{array}{c} \longrightarrow (x) \\ \longleftarrow \text{John} \end{array}$$
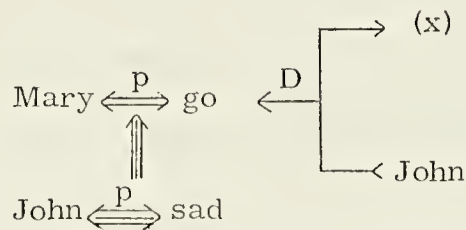
Another example is the sentence "Sam flew his plane to New York". The network for this sentence is

$$\begin{array}{c} \text{Sam} \overset{p}{\Longleftrightarrow} \text{do} \\ \Uparrow \\ \text{plane} \overset{p}{\Longleftrightarrow} \text{fly} \overset{D}{\longleftarrow} \begin{array}{c} \longrightarrow \text{New York} \\ \longleftarrow (x) \end{array} \\ \Uparrow \text{POSS} \\ \text{Sam} \end{array}$$
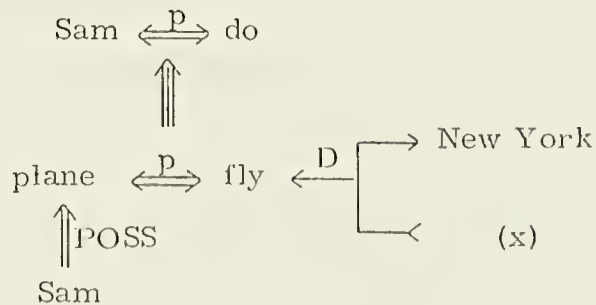
The "do" in the network is a dummy standing for an ACT which was not stated--in this case, the actions which are required to pilot a plane.

11. Time of Conceptualization

This rule relates another conceptual category T (for concepts like "yesterday" and "3 o'clock") and a conceptualization. The time of something modifies the entire conceptualization and not any particular item in it, thus it is the time of the joining together of the actor and the ACT.

$$\text{yesterday}$$
$$\downarrow$$
$$\text{John} \xleftrightarrow{\ p\ } \text{hit} \xleftarrow{\ o\ } \text{Mary}$$
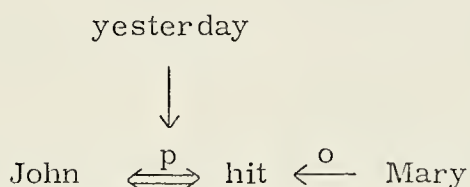
12. Concurrent Conceptualizations

This rule is similar to rule 11, except that here an entire conceptualization is the time of another. This dependency is usually realized in English by "while". "Mary cried while John hit her" yields the conceptual network

$$\text{John} \xleftrightarrow{\ p\ } \text{hit} \xleftarrow{\ o\ } \text{Mary}$$
$$\downarrow$$
$$\text{Mary} \xleftrightarrow{\ p\ } \text{cry}$$
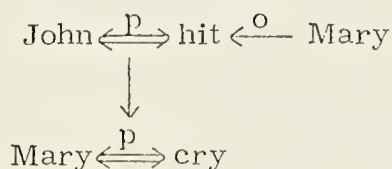
13. Location of Conceptualization

Rule 13 indicates that conceptualizations must occur some-place. For example, "Mary hit John in the park" yields

$$\text{park}$$
$$\Downarrow$$
$$\text{Mary} \xleftrightarrow{\ p\ } \text{hit} \xleftarrow{\ o\ } \text{John}$$
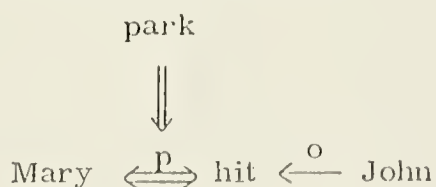
14. Change of State

The final rule expresses the fact that an object in the world has changed its state in some way. The sentence "John grew the plants" would yield the network

$$John \xLeftrightarrow{p} do$$

(where $y > x$)
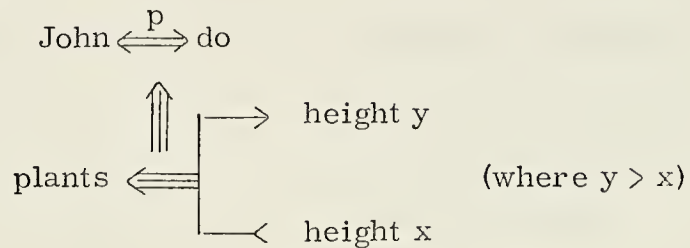
("Grow" is a state change, and not an action that someone can perform, hence the dummy "do".)
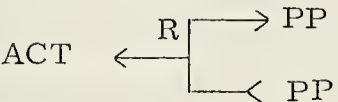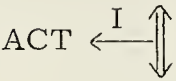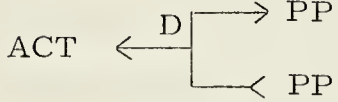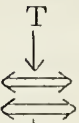
1	PP $\Longleftrightarrow$ ACT	Actor-ACT Dependency
2	PP $\Longleftarrow$ PA	Attributive Predication
3	PP $\Longleftarrow$ PP	ACT Membership
4	PP $\longleftarrow$ PA	Attributive Dependency
5	PP $\Longleftarrow$ PP	Location, Possession, Containment
6	ACT $\overset{o}{\longleftarrow}$ PP	Objective Dependency

7   ACT $\overset{R}{\longleftarrow}$ ⌐→ PP   ⌙↽ PP   Recipient Case

8   ACT $\overset{I}{\longleftarrow}$ ⇕   Instrumental Case

9   ACT $\overset{D}{\longleftarrow}$ ⌐→ PP   ⌙↽ PP   Directive Case

10    $\Longleftrightarrow$ ⇑ $\Longleftrightarrow$ Causality

11    $\overset{T}{\downarrow}$ $\Longleftrightarrow$ Time of Conceptualization

12    $\Longleftrightarrow$ $\downarrow$ $\Longleftrightarrow$ Concurrent Conceptualizations

13    $\overset{PP}{\Downarrow}$ LOC $\Longleftrightarrow$ Location of Conceptualization

14   PP $\Longleftarrow$ ⌐→ PA   ⌙↽ PA   Change of State

Figure 1. CONCEPTUAL RULES

III. NLP--NATURAL LANGUAGE PROCESSOR

NLP is a general language processing system consisting of an IBM 360 Fortran program and a "rule language." Sets of "decoding" and "encoding" rules must be written to specify how processing is to be done for a given application. Decoding rules specify how input text is to be processed to produce an entity-attribute-value data structure (called an Internal Data Structure--IDS), and encoding rules specify how to produce output text from the IDS.

Several of the Fortran routines act as a "compiler" for the rule language. They are used to process decoding and encoding rules and convert them into their internal representation. Other routines, which are used at execution time, perform encoding and decoding according to this information.

A complete description of NLP is found in Ref. 1. This section presents an overview of NLP, with some portions coming directly from Ref. 1.

A. ENTITY-ATTRIBUTE-VALUE STRUCTURE

The IDS is designed to hold information (produced from input text) in a language-independent form. All of the information is held in "records", which are just lists of attribute-value pairs. Some records represent physical entities, such as "car", "ball", or "man". Others represent abstract entities such as actions. There is a record for each

word in the structure, and the values of the attributes of the record specify the characteristics of the word. For example, the record for the word "arrive" could have a part-of-speech attribute with the value "verb", and a type attribute with the value "intransitive". There is also a record for each concept in the structure. The attribute which relates concepts to other concepts is called the SUPerset attribute. For example, the SUP attribute of a record representing the concept "car" might point to another record representing the concept "vehicle," and the SUP of vehicle might be "entity." This SUP chain specifies that "car" is a type of "vehicle," and "vehicle" is a type of "entity."

The use of records is not limited to the representation of just single words and concepts, however. A record may describe the information contained in an entire "conceptualization." For example, the sentence "John gave a book to the girl." could be described by the record

```
SUP          give
AGENT        John
GOAL         book
RECIPIENT    girl
PAST
```

There are many attributes which action records such as these might have, to hold the information present in a conceptualization. These include AGENT, GOAL, LOCATION, TIME, SOURCE, DESTINA-TION, DONOR, RECIPIENT, REASON, INSTRUMENT, and CONCUR-RENT. The values of some of these are entities (i.e., PP's in Con-ceptual Dependency terminology), and the values of some are other

action records. Entities are represented by records which have attributes such as LOCATION, COLOR, SIZE, QUANTITY, WEIGHT, SPEED, OWNER, and CONTAINER.

A special type of attribute used in NLP is the "indicator," so called because it indicates the existence of a particular condition. For example, a record describing the word "cars" could be said to have its PLURal indicator "on." In the example record given above, PAST is an indicator.

Most of the information about words and concepts is initially entered into the system by means of named record definitions. A typical named record definition is

TAK('EVENT', E, ES, ING, EN, ER, TRANS)

This example defines a record named "TAK" which has a SUP attribute pointing to the named record 'EVENT' and has the indicators E, ES, ING, EN, and ER, indicating that 'TAK' can have these endings. It also has the indicator TRANS, meaning that 'TAK' is a transitive verb. All these indicators provide information about the word "take", while the SUP attribute provides information about the concept "take".

The most important feature of named records is that they may be referred to by name both from other named records and from the encoding and decoding rules.

B. THE RULE LANGUAGE

The purpose of NLP, basically, is to convert information from one form to another. The two different forms of information that it

works with are records and character strings. The process of converting character strings into a record structure representing the meaning of the input is called "decoding". The inverse process is called "encoding".

The processing done in the system is specified by sets of rules written in a rule language. Decoding rules specify how input character strings are to be converted to records, and encoding rules specify how records are to be converted to character strings. Either kind of rule can be used to specify how records can be converted to other records. For the application being reported on in this thesis, encoding rules are used to specify all of the processing. A complete explanation of the rule language and description of the application of the rules is given in Ref. 1.

A rule consists of a left part and a right part, separated by an arrow (-->). In general, the left part specifies what conditions must exist in order to apply the rule, and the right part tells what to do when the rule can be applied.

As typical examples of the rules written in the NLP rule language, consider the following:

(1)   VERB('BE')   VERBPH(PASTPART)   -->

                VERBPH(PASSIVE, VFORM=VFORM(VERB))

(2)   VERBPH(PASSIVE) -->

                VERB('BE', VFORM=VFORM(VERBPH))

                VERBPH(-PASSIVE, -VFORM, PASTPART)

The first example is a decoding rule. It says that any form of the verb "be" can be put together with a past-participle verb-phrase to create a new verb-phrase that has all the characteristics of the old verb-phrase, except that it is passive and has the same verb-form (e.g., present-third-person-singular) as the verb on the left. For example, this rule would apply to the phrase "is unloaded".

The second rule is an encoding rule, and says that a passive verb-phrase is to be expanded to a verb which is a form of "be" (with the particular verb-form coming from the verb-phrase), followed by a new verb-phrase which has all the characteristics of the old verb-phrase, except that it is not passive and it has past-participle in place of its old verb-form. The use of encoding rules will be seen in the next section.

IV. IMPLEMENTATION

This section describes the information structure devised for representing a Conceptual Dependency conceptualization in NLP and describes the processing required to convert a conceptualization from its usual NLP form into this form and vice versa.

A. THE INFORMATION STRUCTURE

A conceptualization is represented in the Internal Data Structure of NLP as a set of records representing the concepts of that conceptualization. The action is considered to be the most important building block in the structure. Consequently, there is a special record ('ACTNLIST') in which there are attributes pointing to each action record. Each of these action records has attributes linking the action to the other concepts in the conceptualization. Figure 2 shows a graphic portrayal of the IDS representation for the conceptualization "John gave Mary a red book," and the records R1 and R2 that would be created by NLP during the decoding of the English sentence. The other records would have been part of the named records already in the system. Additionally, R1 would be pointed to by 'ACTNLIST'.

In the work done here it was decided to represent each relation in a conceptualization of Conceptual Dependency theory by an NLP record with specified attributes. For example, a record representing a rule 1 relation would have a SUP attribute of 'REL1' and attributes

Figure 2. INTERNAL DATA STRUCTURE
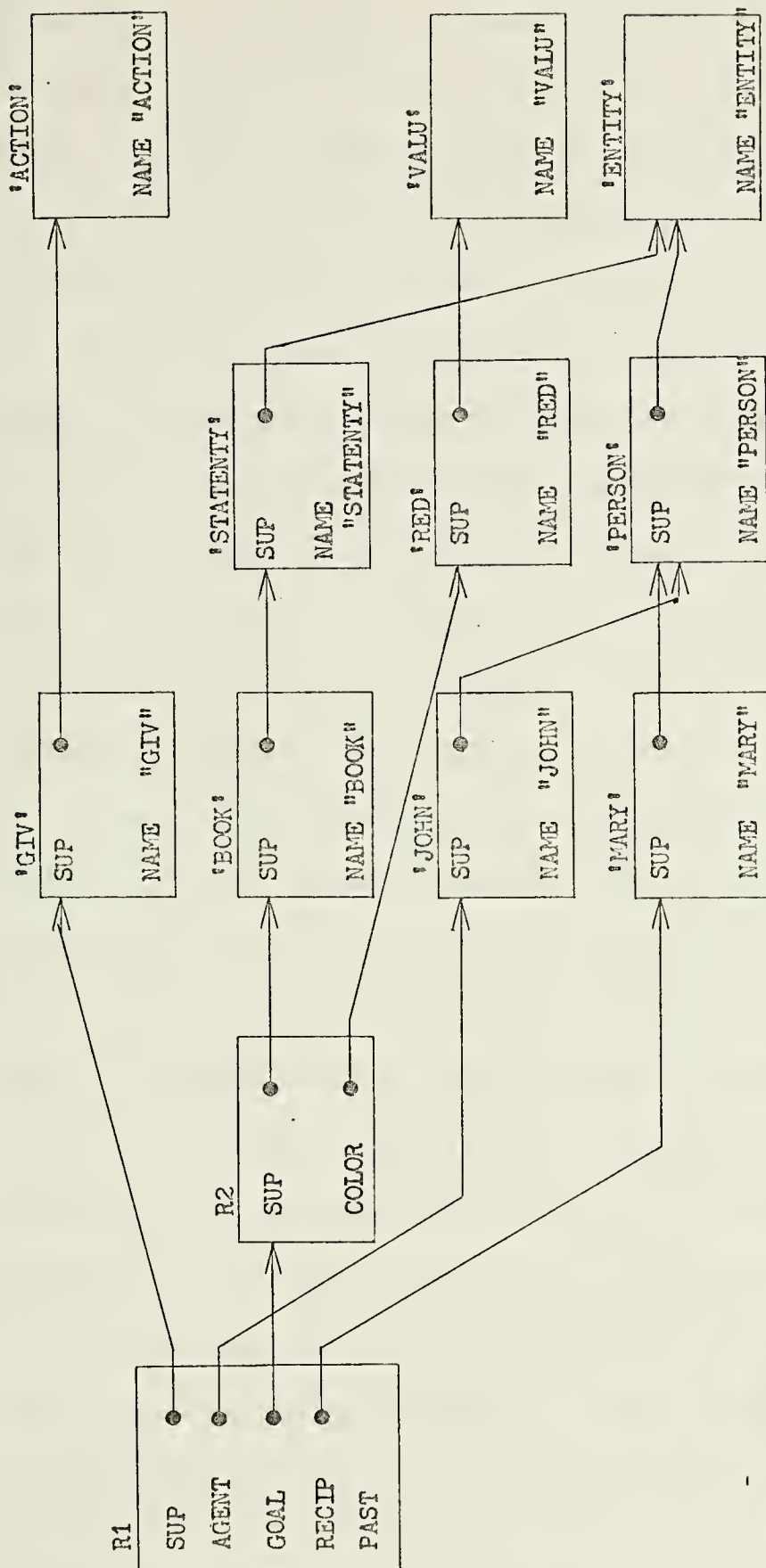
NAMED RECORDS

R1('GIV', AGENT='JOHN', GOAL=R2, RECIP='MARY', PAST)
R2('BOOK', COLOR='RED')

PP and ACT to point to the associated concepts. Similarly, a rule 7 relation yields a 'REL7' with attributes of ACT, PPDON, and PPREC. Figure 3 shows the dependency relations and the attributes specified for each relation. In addition to these attributes, certain other attributes defined in NLP may be used as necessary (e.g., PAST, PLUR).

The three parts of rule 5 (LOCation, CONTainment, and POSSession) were defined separately because it was felt that the preposition associated with location is extremely important to the conceptualization, whereas it is usually understood in the case of possession and containment. For example, the sentence "John was sitting at the bar" takes on entirely different meanings when the location preposition "at" is replaced by "behind", "on", "above", or "under."

Figure 4 shows the representation of an example conceptualization in the structure developed for the dependency relations. The conceptualization shown there is the same as that in Figure 2. The record REL-LIST has pointers to each of the REL records, and the attribute LASTREC says that attribute 14 (@14) is the final pointer attribute on the list. The SUP of each REL record indicates which conceptual relation the record represents, and the other attributes point to the associated concepts. In the figure, records R1, R2, R3, and R4 represent the concepts involved. The SUP attributes of records D1-D5 point to the same named records which are shown in Figure 2 ('BOOK', 'JOHN', 'MARY', etc.). These named records were left off this drawing for lack of space.
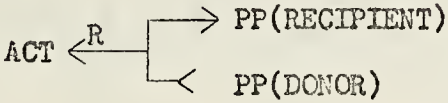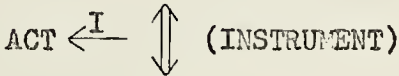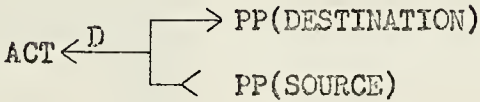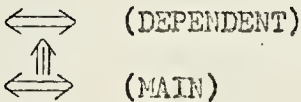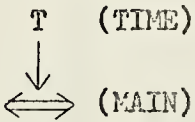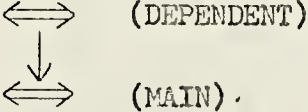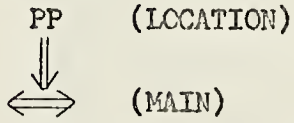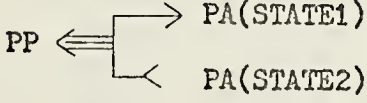
RULE	RELATION	DEFINING ATTRIBUTES
1	PP ⟺ ACT	'REL1', PP, ACT
2	PP ⟺ PA	'REL2', PP, PA
3	PP(MAIN) ⟺ PP(DESCRIPTOR)	'REL3', PPMAIN, PPDESC
4	PP ⟵ PA	'REL4', PP, PA
5A	PP(MAIN) ⟸ PP(LOCATION)	'REL5A', PPMAIN, PPLOC, LOCPREP
5B	PP(MAIN) ⟸ PP(CONTAIN)	'REL5B', PPMAIN, PPCONT
5C	PP(MAIN) ⟸ PP(OWNER)	'REL5C', PPMAIN, PPOWN
6	ACT ⟵ᴼ PP(OBJECT)	'REL6', ACT, PPOBJ
7	ACT ⟵ᴿ ⟶ PP(RECIPIENT) / ⟨ PP(DONOR)	'REL7', ACT, PPDON, PPREC
8	ACT ⟵ᴵ ⇕ (INSTRUMENT)	'REL8', ACT, INSTREL
9	ACT ⟵ᴰ ⟶ PP(DESTINATION) / ⟨ PP(SOURCE)	'REL9', ACT, PPSRC, PPDES
10	⟺ (DEPENDENT) ⇑ ⟺ (MAIN)	'REL10', MAINREL, DEPREL
11	ᵀ (TIME) ↓ ⟺ (MAIN)	'REL11', MAINREL, TIME
12	⟺ (DEPENDENT) ↓ ⟺ (MAIN)	'REL12', MAINREL, DEPREL
13	ᴾᴾ (LOCATION) ⇓ ⟺ (MAIN)	'REL13', MAINREL, PPLOC, LOCPREP
14	PP ⟸ ⟶ PA(STATE1) / ⟨ PA(STATE2)	'REL14', PP, STATE1, STATE2

Figure 3. RELATION RECORD DEFINITIONS

28

The diagram shows boxes labeled D1 through D5 (with SUP values 'TRANS', 'JOHN', 'MARY', 'BOOK', 'RED'), R1 through R4 (with SUP values 'REL1', 'REL6', 'REL7', 'REL4'), and a 'RELLIST' box.

'RELLIST'

LASTREC	14
@11	
@12	
@13	
@14	
NAME "RELLIST"

D1 — SUP 'TRANS'
D2 — SUP 'JOHN'
D3 — SUP 'MARY'
D4 — SUP 'BOOK'
D5 — SUP 'RED'

R1 — SUP 'REL1', ACT, PP, PAST
R2 — SUP 'REL6', ACT, PPOBJ
R3 — SUP 'REL7', ACT, PPDON, PPREC
R4 — SUP 'REL4', PP, PA

NAMED RECORDS

RELLIST (LASTREC=14, @11='R1', @12='R2', @13='R3', @14='R4')

R1 ('REL1', ACT='D1', PP='D2', PAST)
R2 ('REL6', ACT='D1', PPOBJ='D4')
R3 ('REL7', ACT='D1', PPDON='D2', PPREC='D3')
R4 ('REL4', PP='D4', PA='D5')
D1 ('TRANS')
D2 ('JOHN')
D3 ('MARY')
D4 ('BOOK')
D5 ('RED')

Figure 4. RELATION STRUCTURE

29

B. ATTRIBUTE-TO-RELATION PROCESSING

This section deals with the conversion of an NLP entity-attribute-value structure to a relation structure of Conceptual Dependency theory. The NLP encoding rules which do the processing are listed in Fig. 5.

The basic reasoning underlying the development of these rules was that the dependency relations could be determined by examining the records in the NLP representation of a conceptualization. For example, all of the relations involving ACT's could be constructed using action records, which are all conveniently referenced in the single record 'ACTNLIST'. The rest of the relations involve either PP's or PA's. The PP's are available on one of two lists--'MOBLIST' or 'STALIST' (lists constructed by NLP during decoding which point to all entities, both MOBile and STAtionary). The PA's can be found as attributes of the records pointed to by one of the three lists mentioned.

In general, the processing is done as follows: a record is taken from the stack of records to be processed, and examined for attributes needed to construct relation records. If one is found, the appropriate REL record is built and put on the 'RELLIST'. The original record, minus the attribute that caused the creation of this REL record, is put back on the top of the stack to be processed again for other possible relations. By erasing these attributes as relations are found, multiple processing by a given rule is avoided. The records must be put back on the stack if they have not been "examined" by all of the rules, because they may represent more than one relation. When a record is

```
(1)   ATTRTOREL      -->      RECLISTC(%='ACTNLIST', LIST='ACTNLIST', LC=1])
                             RECLISTC(%='MOBLIST', LIST='MOBLIST', LC=1])
                             RECLISTC(%='STALIST', LIST='STALIST', LC=1])

(2)   RECLISTC(LC.LE.LASTREC)  -->
                RECC(%@LC(RECLISTC))(RECLISTC),LIST(RECLISTC))
                LC(RECLISTC)  ENTY=@LC(RECLISTC)(RECLIST2))
                RECLISTC(LC=LC + 1)

(3)   RECLISTC      -->      NULL

(4)   RECC(LIST.EQ.'ACTNLIST')   -->      ACTRECC(%RECC)
(5)   RECC(LIST.EQ.'MOBLIST')    -->      ENTRECC(%RECC)
(6)   RECC(LIST.EQ.'STALIST')    -->      ENTRECC(%RECC)

(7)   ACTRECC(DONOR)  -->
        RELREC('REL7', SUP(ENTY(ACTRECC))='TRANS',ACT=ENTY(ACTRECC),
        PPDON=DONOR(ENTY(ACTRECC)),PPREC=AGENT(ENTY(ACTRECC)))
        ACTRECC(-DONOR,-DONOR(ENTY))

(8)   ACTRECC(RECIP)  -->
        RELREC('REL7', SUP(ENTY(ACTRECC))='TRANS',ACT=ENTY(ACTRECC),
        PPDON=AGENT(ENTY(ACTRECC)),PPREC=RECIP(ENTY(ACTRECC)))
        ACTRECC(-RECIP,-RECIP(ENTY))

(9)   ACTRECC(LOCATION,'ARRIV'|'ENTER')  -->
        RELREC('REL9', ACT=ENTY(ACTRECC),
        PPSRC=SOURCE(ENTY(ACTRECC)),
        PPDES=LOCOBJ(LOCATION(ENTY(ACTRECC))),
        ACTRECC(-LOCATION,-LOCATION(ENTY))

(10)  ACTRECC(SOURCE,'COM'|'GO')  -->
        RELREC('REL9', ACT=ENTY(ACTRECC),
        PPSRC=SOURCE(ENTY(ACTRECC)),
        PPDES=DESTIN(ENTY(ACTRECC)))
        ACTRECC(-SOURCE,-SOURCE(ENTY))
```

FIGURE 5. ATTRIBUTE-TO-RELATION RULES (1 OF 4)

31

(11) ACTRECC(DESTIN,'COM.!'GO') -->
          RELREC('REL9', ACT=ENTY(ACTRECC),
                 PPSRC=SOURCE(ENTY(ACTRECC)),
                 PPDES=DESTIN(ENTY(ACTRECC)))
          ACTRECC(-DESTIN, -DESTIN(ENTY))

(12) ACTRECC(LOCATICN,'LEAV') -->
          RELREC('REL9', ACT=ENTY(ACTRECC),
                 PPSRC=LOCOBJ=LOCATION(ENTY(ACTRECC))),
                 PPDES=DESTIN(ENTY(ACTRECC)))
          ACTRECC(-LOCATION, -LOCATION(ENTY))

(13) ACTRECC(AGENT) -->
          RELREC('REL1', ACT=ENTY(ACTRECC), PP=AGENT(ACTRECC),
                 ONEPTR(ACTRECC)=SEG, VERBPHIND=VERBPHIND(ACTRECC))
          ACTRECC(-AGENT, -AGENT(ENTY), -VERBPHIND(ENTY))

(14) ACTRECC(GOAL) -->
          RELREC('REL6', ACT=ENTY(ACTRECC), PPOBJ=GOAL(ACTRECC))
          ACTRECC(-GOAL, -GOAL(ENTY))

(15) ACTRECC(INST) --> RELREC('REL8', ACT=ENTY(ACTRECC),
                 INSTREL=ONEPTR(INST(ACTRECC)),
                 BACK8(INST(ACTRECC))=SEG)
          ACTRECC(-INST, -INST(ENTY))

(16) ACTRECC(BACK8) -->
          ACTRECC(INSTREL(BACK8(ACTRECC))=ONEPTR(ACTRECC), -BACK8)

(17) ACTRECC(REASCN) -->
          RELREC('REL10', MAINREL=ONEPTR(ACTRECC),
                 DEPREL=ONEPTR(REASON(ACTRECC)),
                 BACK10(REASON(ACTRECC))=SEG)
          ACTRECC(-REASON, -REASON(ENTY))

(18) ACTRECC(BACK10) -->
          ACTRECC(DEPREL(BACK10(ACTRECC))=ONEPTR(ACTRECC), -BACK10)

FIGURE 5. ATTRIBUTE-TO-RELATION RULES (2 OF 4)

(19) ACTRECC(TIME) -->
     RELREC('REL11', MAINREL=ONEPTR(ACTRECC),TIME=TIME(ACTRECC))
     ACTRECC(-TIME, -TIME(ENTY))

(20) ACTRECC(CONCURR) -->
     RELREC('REL12', MAINREL=ONEPTR(ACTRECC),
       DEPREL=ONEPTR(CONCURR(ACTRECC)),
      -DEPREL,BACK12(CONCURR(ACTRECC))=SEG)
     ACTRECC(-CONCURR, -CONCURR(ENTY))

(21) ACTRECC(BACK12) -->
     ACTRECC(DEPREL(BACK12(ACTRECC))=ONEPTR(ACTRECC),-BACK12)

(22) ACTRECC(LOCATION) -->
     RELREC('REL13', MAINREL=ONEPTR(ACTRECC),
       LCCPREP=SUP(LOCATION(ENTY(ACTRECC)),
       PPLOC=LOCOBJ(LOCATION(ENTY(ACTRECC))))
     ACTRECC(-LOCATION, -LOCATION(ENTY))

(23) ACTRECC --> NULL

(24) ENTRECC(SIZE$'RELSIZ') -->
     RECORD(SUP=SIZE(ENTRECC), RP(MEM)=SEG)
     RELREC('REL4', PP=ENTY(ENTRECC), PA=RP(MEM), -RP(MEM))
     ENTRECC(-SIZE, -SIZE(ENTY))

(25) ENTRECC(COLOR) -->
     RECORD(SUP=COLOR(ENTRECC), RP(MEM)=SEG)
     RELREC('REL4', PP=ENTY(ENTRECC), PA=RP(MEM), -RP(MEM))
     ENTRECC(-COLOR, -COLCR(ENTY))

(26) ENTRECC(QUANTITY) -->
     RECORD(SUP=QUANTITY(ENTRECC), RP(MEM)=SEG)
     RELREC('REL4', PP=ENTY(ENTRECC), PA=RP(MEM), -RP(MEM))
     ENTRECC(-QUANTITY, -QUANTITY(ENTY))

FIGURE 5. ATTRIBUTE-TO-RELATION RULES (3 OF 4)

33

(27) ENTRECC(WEIGHT) -->
      RECORD(SUP=WEIGHT(ENTRECC), RP(MEM)=SEG)
      RELREC("REL4", PP=ENTY(ENTRECC), PA=RP(MEM), -RP(MEM))
      ENTRECC(-WEIGHT, -WEIGHT(ENTY))

(28) ENTRECC(SPEED) -->
      RECORD(SUP=SPEED(ENTRECC), RP(MEM)=SEG)
      RELREC("REL4", PP=ENTY(ENTRECC), PA=RP(MEM), -RP(MEM))
      ENTRECC(-SPEED, -SPEED(ENTY))

(29) ENTRECC(LOCATION) -->
      RELREC("REL5A", PPMAIN=ENTY(ENTRECC),
         LOCPREP=LOCATION(ENTY(ENTRECC)),
         PPLOC=LOCOBJ(LOCATION(ENTY(ENTRECC))))
      ENTRECC(-LOCATION, -LOCATION(ENTY))

(30) ENTRECC(CONTAIN) -->
      RELREC("REL5B", PPMAIN=ENTY(ENTRECC),
         PPCONT=CONTAIN(ENTY(ENTRECC)))
      ENTRECC(-CONTAIN, -CONTAIN(ENTY))

(31) ENTRECC(OWNER) -->
      RELREC("REL5C", PPMAIN=ENTY(ENTRECC), PPOWN=OWNER(ENTRECC))
      ENTRECC(-OWNER, -OWNER(ENTY))

(32) ENTRECC --> NULL

(33) RELREC --> NULL(LASTREC("RELLIST")=LASTREC("RELLIST")+1,
      @LASTREC("RELLIST")("RELLIST")=RELREC)

FIGURE 5. ATTRIBUTE-TO-RELATION RULES (4 OF 4)

found to have none of the attributes necessary for construction of a relation, it is deleted from the processing stack, and the next record is taken from the top of the stack and processed. The operation of the stack is basic to the encoding process of NLP and is described in detail in Ref. 1.

The first three processing rules shown in Fig. 5 result in copies of the records in the three lists ('ACTNLIST', 'MOBLIST', and 'STALIST') being placed on the processing stack. Additionally, each of these records is given another attribute ('ENTY') which points to the original record. Rules 4, 5, and 6 cause each record, as it comes off the stack, to be classified as an action record (ACTRECC) or an entity record (ENTRECC). Rules 7 through 30 recognize the presence of dependency relations and create appropriate RELation records (RELREC), and rule 31 causes the newly created RELRECs to be placed on the list 'RELLIST'.

1. Rule Processing Example

To illustrate the application of the processing rules, consider the sentence

"John ate a red apple."

The following records would be present in the IDS as a result of the decoding of this sentence by NLP:

R1 ('EAT', AGENT='R2', GOAL='R3', PAST)

R2 ('JOHN')

R3 ('APPLE', COLOR='RED')

35

Since R1 is an action record, it would be pointed to by 'ACTNLIST'. R2 would be on 'MOBLIST', and 'STALIST' would have an attribute pointing to R3. Assume that each list has just the one record in it, pointed to by attribute 11 (@11) of the list, and that attribute LASTREC of each list is also 11. Processing would proceed as follows. Rule 1 would create 3 records, each named RECLISTC. The first would consist of a copy (%) of 'ACTNLIST', an attribute called LIST pointing to the list 'ACTNLIST', and an attribute LC set to 11. These three records would be placed on the processing stack in inverse order, i.e., the first created is on top, the second one created is next, and so on. (If more than one record is created by any given rule, the records are always put on the stack in this manner.) The stack now consists of

STACK

S1   RECLISTC(@11='R1', LASTREC=11, LIST='ACTNLIST', LC=11)
S2   RECLISTC(@11='R2', LASTREC=11, LIST='MOBLIST', LC=11)
S3   RECLISTC(@11='R3', LASTREC=11, LIST='STALIST', LC=11)

(The designators S1, S2, and S3 are not a part of the rules--they are being used here to indicate the order of creation of the rules.)

The next step in processing is to take S1 off the stack and examine it to see which of the rules applies to it. Rules 2 and 3 both apply to records with the name 'RECLISTC'. Rule 2 says that if a 'RECLISTC' record has an attribute LC which is less than or equal in value to the LASTREC attribute, then execute the "creation specifications" on the right of the arrow (-->). Since, in S1, LC is equal to

LASTREC, the condition is satisfied, so the rule is executed, resulting in a RECC record.

The first creation element is

%@LC(RECLISTC)(RECLISTC)

This says "copy the record pointed to by attribute (LC(RECLISTC)) of RECLISTC," where "RECLISTC" refers to the record being processed (S1, in this case). LC of S1 is 11; attribute 11 of S1 is 'R1', so the RECC begins as a copy of 'R1'. The second creation element says to copy the LIST attribute of the RECLISTC into this record, and the third copies the LC attribute from S1. The last specification says to set the ENTY attribute in this record equal to the value of @LC(RECLISTC) (RECLISTC). Again, LC(S1) is 11, and @11 of S1 is 'R1', so the ENTY attribute of RECC points to R1.

Rule 2 also says to put the current RECLISTC record back on the stack and increment its LC attribute. At this point, the stack consists of

```
S4   RECC('EAT', AGENT='R2', GOAL='R3', PAST, LIST=
         'ACTNLIST', LC=11, ENTY='R1')
S1   RECLISTC(@11='R1', LASTREC=11, LIST='ACTNLIST',
         LC=12)
S2   RECLISTC(@11='R2', LASTREC=11, LIST='MOBLIST',
         LC=11)
S3   RECLISTC(@11='R3', LASTREC=11, LIST='STALIST',
         LC=11)
```

Next, S4 is taken from the stack, and the rules are searched to find one that applies to a "RECC" record. The rules are always searched in order, so that the first one that applies will be used. It is seen in Fig. 5 that rules 4, 5, and 6 are the only possible rules that can be

37

applied at this point. The condition that must be met for rule 4 to be applicable is that the LIST attribute of the RECC be equal to 'ACTNLIST', and this condition is satisfied in S4, so a record named ACTRECC is created as a copy of the current RECC and put on the stack:

```
S5   ACTRECC('EAT', AGENT='R2', GOAL='R3', PAST,
                  LIST='ACTNLIST', LC=11, ENTY='R1')
S1   (same as before)
S2   (same as before)
S3   (same as before)
```

As was mentioned earlier, the purpose of the first six rules is to construct ACTRECCs and ENTRECCs that can be examined for dependency relations. It was also stated that ACTRECCs will yield those relations involving ACTs, and that any relations not involving ACTs must come from the ENTRECCs. The rest of the rules in Fig. 5 are divided into two groups: those that process ACTRECCs and those that process ENTRECCs. The application of these rules will be illustrated by continuing with the example.

The ACTRECC (S5) is removed from the stack and the rules are searched (from the beginning) for one in which the condition specification (if any) is satisfied. The first rule that processes ACTRECCs is rule 7, but it specifies that the current ACTRECC must have a DONOR attribute. (Asking if a particular attribute "exists" means "does this attribute have a value other than zero?") Since S5 does not "have" a DONOR attribute, the rule search is continued until rule 13 is reached. S5 _does_ have an AGENT attribute, so this rule applies. A record called RELREC is created with four attributes. The first is a SUP attribute of 'REL1' indicating that this new record represents a

type 1 dependency relation (PP$\Longleftrightarrow$ACT). The ACT attribute is set equal to the ENTY attribute of the current ACTRECC by the creation element "ACT=ENTY(ACTRECC)", and the PP attribute is set to the AGENT attribute of S5. The fourth creation element, "ONEPTR (ACTRECC)=SEG", says to set the ONEPTR attribute of the current ACTRECC to point to the record being created, i.e., this RELREC. (The reason for this "back-pointer" will become apparent when rules 15 and 16, are discussed.) The final creation element says to make the VERBPHIND ("verbphrase indicator") attribute of this RELREC equal to the VERBPHIND of the ACTRECC. The VERBPHIND includes all indicators, such as PAST, FUTURE, SING, and PLUR, that describe verbs. In this example, the only one of these indicators that is "on" in S5 is PAST, so PAST is set in RELREC. (If none of the indicators of this form had been "on", i.e., VERBPHIND was zero, then the VERBPHIND in this RELREC would have been set equal to zero, also. In that case, it would be said that there was "no" VERB-PHIND indicator present.)

This completes the creation of the RELREC. The other part of this rule "ACTRECC(-AGENT, -AGENT(ENTY), -VERBPHIND (ENTY))" says to "erase" the AGENT attribute of the current ACTRECC (i.e., set it to zero), and erase the AGENT and VERBPHIND attributes of the record pointed to by the ENTY attribute of this ACTRECC. ENTY of S5 points to R1, so now R1 looks like this:

R1 ('EAT', GOAL='R3')

The RELREC that was just created and the "new version" of S5 are

returned to the stack:

    S6   RELREC ('REL1', ACT='R1', PP='R2', PAST)
    S5   ACTRECC ('EAT', GOAL='R3', PAST, LIST='ACTNLIST',
                  LC=11, ENTY='R1', ONEPTR='S6')
    S1   (Same as before)
    S2   (Same as before)
    S3   (Same as before)

It should now be apparent why "AGENT" was erased from S5: the next

time it came off the stack with an AGENT attribute, it would be proc-

essed by rule 13 again, and put back on the stack--an "infinite loop."

The reason for erasing the AGENT attribute from R1 will be seen later.

The next step in the process is to remove S6 from the stack

and find an applicable rule. The only rule that handles a RELREC is

rule 33, and since there are no conditions to be met before processing,

rule 33 is executed. The "NULL" on the right says that after executing

the creation elements (if there are any), do not put the current record

back on the stack.

The creation elements in rule 33 say to increment the LAST-

REC attribute of the record 'RELLIST' (it was set to the value of the

last attribute being used), make this new attribute point to the current

RELREC, S6, and increment LASTREC. It was mentioned earlier in

the discussion that 'RELLIST' is used as a list to keep track of the

relation records.

The next record on the stack is S5, which is taken off to

become the "current" record. This time, in the rule search, the

condition specification of rule 13 is not satisfied, since S5 no longer

has an AGENT attribute. However, S5 does have a GOAL attribute, so

rule 14 is executed, resulting in a RELREC with a SUP of 'REL6', an

ACT pointing to R1, and a PPOBJ pointing to the GOAL of the ACTRECC,

R3. (This RELREC represents the relation EAT $\xleftarrow{O}$ APPLE.) Then the

GOAL attributes are erased from S5 and R1, and the 'REL6' just created

is put on the stack, along with S5.

```
S7   RELREC ('REL6', ACT='R1', PPOBJ='R3')
S5   ACTRECC ('EAT', PAST, LIST='ACTNLIST',
              LC=11, ENTY='R1', ONEPTR='S6')
S1   (Same as before)
S2   (Same as before)
S3   (Same as before)
```

S7 is taken off the stack and rule 33 is executed, adding the

new relation record to 'RELLIST'.

When the ACTRECC comes off the stack and the rules are

searched, it is found that none of the ACTRECC rules that specify a

particular condition (rules 7-22) are satisfied, which means that the

current ACTRECC is no longer needed--it has supplied all the informa-

tion it can supply. Rule 23 is executed and takes care of this situation

by not putting the ACTRECC back on the stack. The stack has now been

reduced to

```
S1   RECLISTC (@11='R1', LASTREC=11, LIST='ACTNLIST', LC=12)
S2   RECLISTC (@11='R2', LASTREC=11, LIST='MOBLIST', LC=11)
S3   RECLISTC (@11='R3', LASTREC=11, LIST='STALIST', LC=11)
```

S1 is the next record off the stack. This time rule 2 does not

apply, because LC is greater than LASTREC, so rule 3 is executed and

S1 is deleted from processing. If 'ACTNLIST' had more than one record,

S1 would go back on the stack after the next 'ACTRECC' had been created,

and the looping process would continue until all of the action records

on 'ACTNLIST' had been processed and all of the relations involving

ACTs had been created and added to 'RELLIST'.

S2 would be processed next, and after rules 2 and 5 had

created an ENTRECC, the stack would look like this:

```
S8   ENTRECC ('JOHN', LIST='MOBLIST', LC=11, ENTY='R2')
S2   RECLISTC (@11='R2', LASTREC=11, LIST='MOBLIST', LC=12)
S3   RECLISTC (@11='R3', LASTREC=11, LIST='STALIST', LC=11)
```

,When S8 comes off the stack, it would ignore the first 23 rules

because they are not concerned with ENTRECCs. The next eight rules

would not apply because S8 has none of the attributes they require, and

rule 32 would prevent this ENTRECC from being considered again for

processing.

S2 comes off the stack again, but is discarded because LC is

greater than LASTREC, leaving only S3 to be processed. Record S3

yields an ENTRECC which is placed on the stack:

```
S9   ENTRECC ('APPLE', COLOR='RED', LIST='STALIST',
               LC=11, ENTY='R3')
S3   RECLISTC (@11='R3', LASTREC=11, LIST='STALIST', LC=12)
```

When S9 is removed from the stack and the rules are searched,

it is found to satisfy the condition of rule 25, i.e., it is an ENTRECC with

a COLOR attribute. The first creation specification says to build a

record ('RECORD') with a SUP of 'RED', and set an attribute (RP) in a

record called MEM to point to this 'RECORD'. (For this example,

assume that 'RECORD' is R4.)

Then a RELREC is created with a SUP of 'REL4', a PP equal

to 'R3' (the ENTY of ENTRECC), and a PA attribute equal to RP (MEM),

42

which in turn is pointing to the RECORD just created. The RP(MEM) is now erased, having served its purpose as a temporary pointer. The reason it was needed is that the relation structure utilizes a different form of record for these attributes than does the IDS, as may be seen in Figs. 2 and 4. For example, the IDS uses a <u>named record</u> 'RED', whereas the relation structure needs a record whose <u>SUP</u> attribute points to the named record 'RED', so this record must be created by the rule, hence: R4 ('RED').

Rule 25 then erases the COLOR attribute from S9 and R3, and places S9 back on the stack along with the new RELREC:

```
S10    RELREC ('REL4', PP='R3', PA='R4')
S9     ENTRECC ('APPLE', LIST='STALIST', LC=11, ENTY=
          'R3')
S3     RECLISTC (@11='R3', LASTREC=11, LIST='STALIST',
          LC=12)   .
```

In processing the remainder of the stack, it can be seen that S10 is put on the relation list, and S9 and S3 are both deleted, thus clearing the stack and ending the processing. The following records have been created:

```
RELLIST    (@11='S6', @12='S7', @13='S10', LASTREC=13)
S6         ('REL1', ACT='R1', PP='R2', PAST)
S7         ('REL6', ACT='R1', PPOBJ='R3')
S10        ('REL4', PP='R3', PA='R4')
R4         ('RED')
```

During the processing of the records, each time an attribute was erased from an ACTRECC or ENTRECC, that same attribute was erased from the original record pointed to by the ENTY attribute of the ACTRECC or ENTRECC. As a result the original records now look like this:

43

R1 ('EAT')
R2 ('JOHN')
R3 ('APPLE')

They have been converted to records that express only the concepts

in the original sentence, which is exactly what is needed by the depen-

dency relation structure.

Using S6, S7, and S10 along with the records they reference,

the dependency network can be constructed:

$$R2 \overset{p}{\Longleftrightarrow} R1 \overset{o}{\longleftarrow} R3$$
$$\uparrow$$
$$R4$$
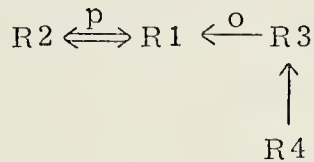
Substituting the "names" of the named records yields:

$$JOHN \overset{p}{\Longleftrightarrow} EAT \overset{o}{\longleftarrow} APPLE$$
$$\uparrow$$
$$RED$$
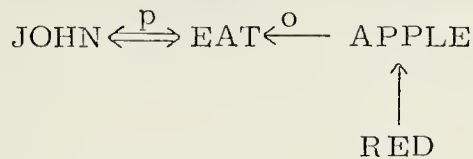
which can be recognized as the same conceptualization represented by

the English sentence at the beginning of this example.

2. Explanation of Remaining Rules

The remainder of this section will deal with the explanation

of the processing rules in Fig. 5 that have not yet been mentioned.

Rules 7 and 8 deal with the recipient case of the dependency

relations. In the discussion of relation 7 a new conceptual ACT

"TRANS" was defined. It was stated that the direction of transition

could be determined by noting the relation between the actor and the

donor or recipient. For the concept "give", the actor and the donor

are the same, and for "take" the actor and the recipient are the same.

However, in an action record, since the agent is already specified, only the donor or recipient need be specified to complete the conceptual relation. So when an ACTRECC satisfies the condition of rule 7 (i.e., has a DONOR), it is known that the AGENT is the RECIPient. The verb is changed to 'TRANS' and the required attributes for a 'REL7' are created: ACT is set to the original record which now has a SUP of 'TRANS'; the PPDON attribute points to the DONOR specified by the ENTY of the current ACTRECC; and PPREC is set to the AGENT. Then the DONOR attribute is cleared from the ACTRECC and its ENTY, and the ACTRECC is placed back on the stack.

Similarly, if the ACTRECC has a RECIP attribute, then rule 8 applies. Here, however, the PPDON is set to AGENT and PPREC is the RECIPient.

Rules 9 and 12 deal with one interpretation of the directive case. Consider the following sentences:

John arrived at the station.

John entered the station.

John left the station.

The verbs in the first 2 sentences cause the location "station" to be thought of as a "destination", while the verb "leave" makes the location appear to be the "source" of travel. Hence, depending on the particular verb, the source and destination of the directive case may need to be interchanged.

In NLP, as has been discussed, the preposition associated with a location is considered to be an important part of the concept, so a

separate record is utilized to hold the preposition. In a record that has a location associated with it, a LOCATION attribute points to a record which has a SUP of the specific preposition involved, and a LOCOBJ attribute pointing to the actual location.

In rule 9, then, the condition that it be applied is that the ACTRECC must have a LOCATION attribute and a SUP of either 'ARRIV' or 'ENTER'. If rule 9 is applicable, it creates a 'REL9' with the required attributes ACT, PPSRC, and PPDES. In this case, the location is considered to be the destination, so PPDES is made to point to the location, which is the value of the LOCOBJ attribute of the record pointed to by LOCATION. PPSRC is given the value of the SOURCE attribute of the action record. Rule 12 is similar to this, but the source and destination are interchanged. PPSRC is set to the location, while PPDES is set to the DESTIN of the action record being processed.

Rules 10 and 11 handle the other interpretation of the directive case--that in which the concept of "motion" is involved.

Rules 15 and 16 process the instrumental case, relation 8. These rules contain some creation elements that have not been encountered before, and are best explained by example. Consider the sentence

John broke the window by throwing a rock.

and the two ACTRECCs (with just the pertinent information) associated with it:

R1 ('BREAK', AGENT='JOHN', GOAL='WINDOW', INST='R2')
R2 ('THROW', AGENT='JOHN', GOAL='ROCK')

Assume for the moment that R2 comes off the stack first (which would be the case if the order of the original action records had been reversed in the 'ACTNLIST'). When it is processed by rule 13 for a 'REL1', a ONEPTR attribute in R2 would be set to point to the 'REL1' record being created, say R3, which now represents the instrument of the conceptualization being considered. R2 would then be processed by rule 14, and then deleted from the processing list (but not from the system).

When R1 comes off, it also is processed by rules 13 and 14, and then rule 15 would be applied. A RELREC would be created, say R4, with a SUP of 'REL8' and an ACT attribute pointing to the ENTY of R1. Then an INSTREL (for INSTrumental RELation) would be set to point to R3, which is the value of the ONEPTR attribute of the record pointed to by INST of R1. That is, in the creation specification

INSTREL=ONEPTR(INST(ACTRECC))

INST(ACTRECC) is R2, and the ONEPTR of R2 points to R3. The next element, "¬INSTREL" (called a "condition on the right"), says that if, at this point, there is no INSTREL attribute in the record, then execute the elements that follow. Otherwise, cease execution of this rule and go on with the processing. Since there is an INSTREL attribute in this record, the rest of the rule would be ignored and processing would continue with R4 being added to 'RELLIST'. It has all the required attributes at this point:

SUP = 'REL8'
ACT = (pointer to ENTY of R1)
INSTREL = 'R3'

47

This expresses the instrumental relation between the ACT "break" and the instrument R3, "John throw rock".

But now, going back to the initial stack, suppose R1 comes off first for processing. It will be processed by rules 13, 14, and 15. Rule 15 would create a RELREC, say R5, with a SUP of 'REL8' and the appropriate ACT attribute. Then the next creation element would set the INSTREL attribute to the ONEPTR of R2. However, since R2 has not yet been processed by rule 13, its ONEPTR attribute is zero, so the INSTREL is zero. This time, when the condition "¬INSTREL" is tested, it yields a value of "true" because, at this point, the INSTREL attribute is <u>not</u> set. Consequently, the next creation element <u>is</u> executed. It says to set a BACK8 attribute in R2 to point to this RELREC, so now R2 is as follows:

R2 ('THROW', AGENT='JOHN', GOAL='ROCK', BACK8='R5')

R5 would be placed on 'RELLIST', but would be incomplete at this point--it still needs an INSTREL attribute. R1 would be deleted from the processing list, and R2 would come off the stack. After being processed by rules 13 and 14, R2 would still have its BACK8 attribute plus a ONEPTR (from rule 13) pointing to the 'REL1' created by rule 13. This 'REL1' represents the instrument of R1. R2 would be picked up by rule 16 since it does have a BACK8 attribute. This rule would set the INSTREL attribute of R5 equal to the ONEPTR of R2 (which is pointing to the "instrument"), and then erase the BACK8 attribute and place this ACTRECC(R2) back on the stack for processing. But now the

conceptualization is complete in the relation structure because the required INSTREL attribute has been set in the previously created RELREC. So by the use of ONEPTR and the "back-pointer" BACK8, the required attribute has been "back-stuffed" into the relation record.

Rules 17 and 18 create the relation record for "causality"-- relation 10, and rules 20 and 21 handle relation 12--that of "concur- rence". Since pointers to other relations are involved, the same problems arise as were discussed for rules 15 and 16. Consequently, the processing is done in an identical manner as that in rules 15 and 16. The only difference is that here there are two 'REL1's involved instead of one 'REL1' and an ACT.

Rule 19 builds a 'REL11', which represents the <u>time</u> of a conceptualization, and rule 22 yields a 'REL13', which gives the <u>location</u> of a conceptualization.

The rest of the rules in Fig. 5 are used to process entity records. Rules 24-28 all create 'REL4's, but are separated because of the way NLP handles this type of attribute. The manner in which these rules are applied was described in the example at the beginning of this section. The only thing in these five rules that has not been mentioned before is the condition specification of rule 24, SIZE$'RELSIZ'. This specifies that the SIZE attribute being considered must be in the set ($) 'RELSIZ' ("relative" size, as opposed to "absolute" size).

The final three rules in this group (29-31) deal with the three parts of conceptual relation 5.

Rule 29 is concerned with location, as was rule 22, but here it is the location of an entity, rather than the location of a conceptualization, that is being processed. Again, the location preposition is prominent. In both rules 30 and 31, the preposition is ignored because the meaning is clear.

Before concluding this section, one further remark should be made. It may appear that rules 7-12 are out of order, but this is not the case. They each require information about some attribute in the action record that gets erased in later processing. For example, rules 7 and 8 both require the AGENT attribute, but if rule 13 had processed the record first, there would no longer be an AGENT attribute.

C.   RELATION-TO-ATTRIBUTE PROCESSING

The procedure for converting information from the relation structure form to the entity-attribute-value form is much simpler than the inverse procedure just discussed. There, because of the uncertainty of the content of the input, a rather complicated process was required to obtain all the pertinent information for the construction of the relation records. Since the relation records have a precisely defined content, it is a fairly straightforward process to set the appropriate attributes in the action and entity records. It must be stressed, though, that only simple cases have been considered in this report.

The rules for this processing are shown in Fig. 6. The processing of records here follows the same general pattern as discussed

(34) RELTOATTR   -->   RECLISTD(%"RELLIST", LC=11)

(35) RECLISTD(LC.LE.LASTREC)   -->   RELRECC(%@LC(RECLISTD)(RECLISTD))
                    RECLISTD(LC=LC+1)

(36) RECLISTD   -->   NULL

(37) RELRECC("REL1")   -->
        NULL(AGENT(ACT(RELRECC))=PP(RELRECC),
             VERBPHIND(ACT(RELRECC))=VERBPHIND(RELRECC))

(38) RELRECC("REL2")   -->
        NULL("BE", SUBJECT=PP(RELRECC), PREDADJ=SUP(PA(RELRECC)),
             VERBPHIND=VERBPHIND(RELRECC),
             RELCL(PP(RELRECC))=SEG, SENT(MEM),
             -RELCL(PP(RELRECC)), SENT(MEM)=SEG)

(39) RELRECC("REL3")   -->
        NULL("BE", SUBJECT=PPMAIN(RELRECC),
             PREDNOM=PPDESC(RELRECC),
             VERBPHIND=VERBPHIND(RELRECC),
             RELCL(PPMAIN(RELRECC))=SEG, SENT(MEM),
             -RELCL(PPMAIN(RELRECC)), SENT(MEM)=SEG)

(40) RELRECC("REL4")   -->
        NULL(@ATTRIB($(SUP(PA(RELRECC))))(PP(RELRECC))=SUP(PA(RELRECC)))

(41) RELRECC("REL5A")   -->
        NULL(LOCATION(PPMAIN(RELRECC))=LOCPREP(RELRECC),
             LOCOBJ(LOCATION(PPMAIN(RELRECC)))=PPLOC(RELRECC))

(42) RELRECC("REL5B")   -->
        RECORD(SUP="IN", LOCOBJ=PPCCNT(RELRECC), RP(MEM)=SEG)
        NULL(LOCATION(PPMAIN(RELRECC))=RP(MEM), -RP(MEM))

FIGURE 6. RELATION-TO-ATTRIBUTE RULES (1 OF 2)

51

```
(43)  RELRECC('REL5C')       -->      NULL(OWNER(PPMAIN(RELRECC))=PPOWN(RELRECC))

(44)  RELRECC('REL6')        -->      NULL(GOAL(ACT(RELRECC))=PPOBJ(RELRECC)      -->

(45)  RELRECC('REL7', VERBSW(MEM), AGENT(ACT).EQ.PPDON)      -->
      NULL(SUP(ACT(RELRECC))='GIV', RECIP(ACT(RELRECC))=PPREC)

(46)  RELRECC('REL7', VERBSW(MEM), AGENT(ACT).EQ.PPREC)      -->
      NULL(SUP(ACT(RELRECC))='TAK', DONOR(ACT(RELRECC))=PPDON(RELRECC))

(47)  RELRECC('REL7')        -->
      NULL(DONOR(ACT(RELRECC))=PPDON(RELRECC);
           RECIP(ACT(RELRECC))=PPREC(RELRECC))

(48)  RELRECC('REL8')        -->      NULL(INST(ACT(RELRECC))=INSTREL(RELRECC))

(49)  RELRECC('REL9')        -->
      NULL(SOURCE(ACT(RELRECC))=PPSRC(RELRECC);
           DESTIN(ACT(RELRECC))=PPDES(RELRECC))

(50)  RELRECC('REL10')       -->
      NULL(REASON(ACT(MAINREL(RELRECC)))=ACT(DEPREL(RELRECC)))

(51)  RELRECC('REL11')       -->
      NULL(TIME(ACT(MAINREL(RELRECC)))=TIME(RELRECC))

(52)  RELRECC('REL12')       -->
      NULL(CONCURR(ACT(MAINREL(RELRECC)))=ACT(DEPREL(RELRECC)))

(53)  RELRECC('REL13')       -->
      NULL(LOCATION(ACT(MAINREL(RELRECC)))=LOCPREP(RELRECC);
           LOCOBJ(LOCATION(ACT(MAINREL(RELRECC))))=PPLOC(RELRECC))

(54)  RELRECC('REL14')       -->      NULL(STATE1(PP(RELRECC))=STATE1(RELRECC);
                                           STATE2(PP(RELRECC))=STATE2(RELRECC))

(55)  RECORD      -->   NULL

(56)  NULL        -->   OUTPUT
```

FIGURE 6. RELATION-TO-ATTRIBUTE RULES (2 OF 2)

52

earlier: the records are placed on a stack, then one at a time they are removed. The rules are searched to find the one that is applicable to the current record, and the creation specifications of that rule are executed. The difference at this point is that once the rule has been applied, the record being processed is no longer needed, so does not go back on the stack. This means that each REL record is processed only once.

1. Rule Processing Example

To illustrate this processing, the relation structure for the conceptualization used in the previous section will be processed (referring to Fig. 6). The records representing the structure are:

```
RELLIST (@11='R1', @12='R2', @13='R3', LASTREC=13)
R1 ('REL4',  PP='D6',  PA='D7')
R2 ('REL6',  ACT='D4',  PPOBJ='D6')
R3 ('REL1',  ACT='D4',  PP='D5',  PAST)
D4 ('EAT')
D5 ('JOHN')
D6 ('APPLE')
D7 ('RED')
```

(The record designations have been changed, but the records themselves are identical to the previous ones.)

Rule 34 puts a copy of 'RELLIST' on the stack and sets LC to 11:

```
S1  RECLISTD(@11='R1', @12='R2', @13='R3', LASTREC=13,
          LC=11)
```

S1 comes off the stack and rule 35 puts a RELRECC on the stack (a copy of @11 of RECLISTD) along with S1, which now has LC=12:

```
S2  RELRECC('REL4',  PP='D6',  PA='D7')
S1  RECLISTD(@11='R1', @12='R2', @13='R3', LASTREC=13,
          LC=12)
```

When the top record (S2) comes off the stack, the rules are searched for one in which the condition specifications are satisfied, and rule 40 is found to apply, so the right side of the rule is executed. The rather complicated expression in rule 40 bears some explanation, but first a word about its purpose.

A 'REL4' expresses the relationship between a PP and a PA. Currently, in NLP all PAs are realized as adjectives and, as such, they all represent attributes such as size, color, quantity, etc. The value of an ATTRIB attribute is the name of one of these attributes. For example, consider the named record definitions

RED('ABSCOLR')
ABSCOLR('QUALVAL', ATTRIB='COLOR')

These say that 'RED' is in the set 'ABSCOLR', and 'ABSCOLR' is in the set 'QUALVAL' with an ATTRIB attribute of 'COLOR'.

It can be seen, then, that given a concept such as "red", the type of attribute it represents may be determined by searching its SUP chain until an ATTRIB is found, and the value of ATTRIB will be the name of the attribute that the concept represents. The first part of the right side of rule 40 does precisely this. It says to take the PA of the current RELRECC, get its SUP, and search the SUP chain ($) of that SUP until an ATTRIB attribute is found. Then cause this attribute (@) of the PP of RELRECC to point to the SUP of the PA of RELRECC. Applying this rule to S2, the processing proceeds as follows: The PA of S2 points to D7; the SUP of D7 is 'RED'; searching the SUP chain of 'RED' until an ATTRIB attribute is found yields 'COLOR'; set the

COLOR attribute of D6 to the SUP of D7. Record D6 now looks like this:

D6 ('APPLE', COLOR='RED')

and the stack has been reduced to S1.

Again, rule 35 is applied, yielding

S3   RELRECC('REL6', ACT='D4', PPOBJ='D6')
S1   RECLISTD(@11='R1', @12='R2', @13='R3', LASTREC=13, LC=13)

S3 is a 'REL6', so rule 44 applies. The GOAL attribute of the ACT specified in S3 is set to the PPOBJ of S3, which makes the GOAL attribute of D4 equal to 'D6':

D4 ('EAT', GOAL='D6')

Rule 35 is applied once more, putting a copy of R3 on the stack and incrementing LC of S1 to 14:

S4   RELRECC('REL1', ACT='D4', PP='D5', PAST)
S1   RECLISTD(@11='R1', @12='R2', @13='R3', LASTREC=13, LC=14)

Record S4 is taken from the stack, and rule 37 causes the AGENT of D4 to point to 'D5', then sets the VERBPHIND of D4 to "PAST":

D4   ('EAT', GOAL='D6', AGENT='D5', PAST)

Finally, S1 comes off the stack and can no longer be processed by rule 35 because LC is now greater than LASTREC, so rule 36 is applied, leaving the stack empty and completing the processing.

The action record (D4) and those records that it references (D5 and D6) make up the IDS representation of the conceptualization "John ate a red apple."

2. Explanation of Remaining Rules

The rest of this section describes the remainder of the rules shown in Fig. 6.

Rule 38 processes a 'REL2', which represents the second dependency relation in Fig. 3. The English realization of this relation is either a sentence (e.g., "John is big.") or a relative clause ("John, who is big, . . . "), so rule 2 handles both possibilities. A record is created with a SUP of 'BE', and attributes of SUBJECT, PREDADJ, and VERBPHIND, and the RELCL (relative clause) attribute of the PP is made to point to this record. Next, a check is made on SENT(MEM), an attribute that contains a pointer to a record which contains information for the main clause of a sentence. In this case, if SENT(MEM) has a value, then the PP of RELRECC is already designated to be put out as part of a sentence, so the record just created will be processed as a relative clause modifying that PP. Otherwise, the RELCL attribute of the PP will be cleared and this record will be put out as a sentence by itself ("John is big.").

Rule 39 does the same processing for a 'REL3'--"John is a doctor." or "John, who is a doctor, . . . ".

Rule 41 sets the LOCATION attribute of the PP to point to the preposition record, and the LOCOBJ attribute of the preposition record to the actual location.

In creating the 'REL5B' relation record from the entity record (rule 30), the preposition was ignored because, conceptually,

it is understood. However, the preposition is needed to complete the representation in the IDS, so it must be supplied. Rule 42 does this by creating a record with a SUP of 'IN' and a LOCOBJ attribute pointing to the "container". Then the LOCATION attribute of the main PP is made to point to this new record.

Rule 43 sets the OWNER attribute of the main PP equal to the PPOWN attribute of the 'REL5C', which points to the "possessor" of the main PP.

Rules 45-47 process the 'REL7' relation, or recipient case. The option is given here of either changing the verb to 'GIV' or 'TAK' (whichever is appropriate) or leaving it as 'TRANS'. The option is determined by the setting of the indicator VERBSW(MEM). If it is set, either 'GIV' or 'TAK' will be the action in the action record. Otherwise, the action will be 'TRANS'.

In rules 48 and 50-53, each of the relation records references another relation, but all of these are realized in the entity-attribute-value structure as attributes of an action record. Hence, these rules reference the ACT of the relations.

Rules 49 and 54 simply set attributes in the appropriate records. The final two rules, 55 and 56, dispose of RECORD and NULL records.

D. EXAMPLES OF RULE APPLICATION

This section presents examples of conceptualizations that were processed by the rules described in the two preceding sections. The

rules were compiled and executed by NLP on the CP/CMS time-sharing system of the IBM 360/67. The conceptualizations were input to the system as named record definitions in the form that would be available to the rules if NLP had decoded the sentences.

For example, the data presented to the ATTRIBUTE-TO-RELATION rules was in the attribute-value form of NLP, while that given to the RELATION-TO-ATTRIBUTE rules was in the format defined here for the dependency relation structure.

Figures 7 and 8 show the input to the rules and the output obtained for the conceptualization

"John gave a big red book to the little girl."

The dependency relation network for this conceptualization is
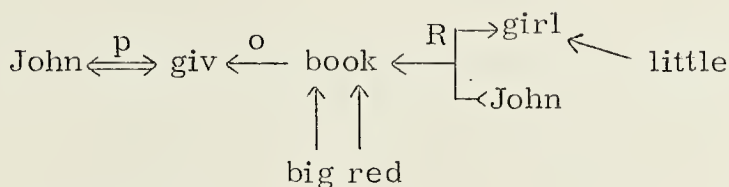
(Note in Fig. 7., that after processing, 'R1' has a SUP attribute of 'TRANS'. Also, in Fig. 8., if VERBSW(MEM) had not been set, 'D2' would have a SUP of 'TRANS'.)

Figure 9 and 10 show the input and output for

"John hit Mary on the swing in the park yesterday while the dog barked"

and its conceptual network.

yesterday   park

$$\text{John} \xleftrightarrow{\ p\ } \text{hit} \xleftarrow{\ o\ } \text{Mary}$$

in

John $\xleftrightarrow{\ p\ }$ hit $\xleftarrow{\ o\ }$ Mary

dog $\xleftrightarrow{\ p\ }$ bark   swing

on
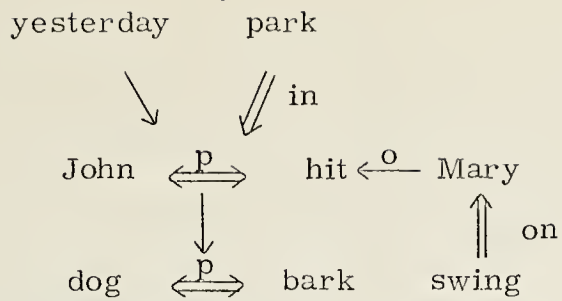
The records in the output of each example that are labeled "C-" are records that were created during processing by the rules. The "C" designations are being used here to represent the actual numerical values given to the records by the system.

INPUT

    INDICATORS:
          VERBPHIND   13-33
                PAST   13,   PLUR   15
          VERBSW   49

    ATTRIBUTES:
          SUP   1

    NAMED RECORDS:
          RELLIST (LASTREC=10)
        ' ACTNLIST (@11='R1', LASTREC=11)
          MOBLIST (@11='R3', @12='R4', LASTREC=12)
          STALIST (@11='R2', LASTREC=11)
          R1 ('GIV', AGENT='R4', GOAL='R2', RECIP='R3',PAST)
          R2 ('BOCK', COLCR='RED', SIZE='BIG')
          R3 ('GIRL', SIZE='LITTLE')
          R4 ('JOHN')
          BIG ('RELSIZ')
          RED ('ABSCOLR')
          LITTLE ('RELSIZ')
          RELSIZ ('RELVAL', ATTRIB='SIZE')
          ABSCOLR ('QUALVAL', ATTRIB='COLOR')

OUTPUT

          RELLIST (@11='C1', @12='C2', @13='C3', @14='C4',
                  @15='C5', @16='C6', LASTREC=16)
          C1 ('REL7', ACT='R1', PPDON='R4', PPREC='R3')
          C2 ('REL1', ACT='R1', PP='R4', PAST)
          C3 ('REL6', ACT='R1', PPOBJ='R2')
          C4 ('REL4', PP='R3', PA='C7')
          C5 ('REL4', PP='R2', PA='C8')
          C6 ('REL4', PP='R2', PA='C9')
          C7 ('LITTLE')
          C8 ('BIG')
          C9 ('RED')
          R1 ('TRANS')
          R2 ('BOOK')
          R3 ('GIRL')
          R4 ('JOHN')

          EXAMPLE OF ATTRIBUTE-TO-RELATION FROCESSING
        "JOHN GAVE A BIG RED BOOK TO THE LITTLE GIRL."

                        FIGURE 7.

60

INPUT

        INDICATORS:
              VERBPHIND   13-33
                    PAST   13,   PLUR   15
              VERBSW   49

        ATTRIBUTES:
              SUP   1

        NAMED RECORDS:
              RELLIST (@11='R1', @12='R2', @13='R3', @14='R4',
                    @15='R5', @16='R6', LASTREC=16, VERBSW(MEM))
              R1 ('REL1', PP='D1', ACT='D2', PAST)
              R2 ('REL6', ACT='D2', PPOBJ='D3')
              R3 ('REL7', ACT='D2', PPDON='D1', PPREC='D6')
              R4 ('REL4', PP='D3', PA='D4')
              R5 ('REL4', PP='D3', PA='D5')
              R6 ('REL4', PP='D6', PA='D7')
              D1 ('JOHN')
              D2 ('TRANS')
              D3 ('BOOK')
              D4 ('BIG')
              D5 ('RED')
              D6 ('GIRL')
              D7 ('LITTLE')
              BIG ('RELSIZ')
              LITTLE ('RELSIZ')
              RED ('ABSCOLR')
              RELSIZ ('RELVAL', ATTRIB='SIZE')
              ABSCOLR ('QUALVAL', ATTRIB='COLOR')

OUTPUT
              D1 ('JOHN')
              D2 ('GIV', AGENT='D1', GOAL='D3', RECIP='D6',PAST)
              D3 ('BOOK', SIZE='BIG', COLOR='RED')
              D6 ('GIRL', SIZE='LITTLE')

              EXAMPLE OF RELATION-TO ATTRIBUTE PROCESSING
           "JOHN GAVE A BIG RED BOOK TO THE LITTLE GIRL."

                            FIGURE 8.

                              61

INPUT

    INDICATORS:
        VERBPHIND 13-33
            PAST 13,   PLUR 15
        VERBSW 49

    ATTRIBUTES:
        SUP   1

    NAMED RECORDS:
        ACTNLIST (@11='R1', @12='R2', LASTREC=12)
        MOBLIST (@11='R3', @12='R4', LASTREC=12)
        STALIST (@11='R6', @12='R8', LASTREC=12)
        R1 ('HIT', AGENT='R3', GOAL='R4', CONCURR='R2',
                TIME='R9', LOCATION='R7', PAST)
        R2 ('BARK', AGENT='R10', PAST)
        R3 ('JOHN')
        R4 ('MARY', LOCATION='R5')
        R5 ('ON', LOCOBJ='R6')
        R6 ('SWING')
        R7 ('IN', LOCOBJ='R8')
        R8 ('PARK')
        R9 ('YESTERDAY')
        R10 ('DOG')

OUTPUT
        RELLIST (@11='C1', @12='C2', @13='C3', @14='C4',
            @15='C5', @16='C6', @17='C7', LASTREC=17)
        C1 ('REL1', ACT='R1', PP='R3', PAST)
        C2 ('REL6', ACT='R1', PPOBJ='R4')
        C3 ('REL11', TIME='R9', MAINREL='C1')
        C4 ('REL12', MAINREL='C1', DEPREL='C6')
        C5 ('REL13', MAINREL='C1', LOCPREP='R7',
                PPLCC='R8')
        C6 ('REL1', ACT='R2', PP='R10', PAST)
        C7 ('REL5A', LOCPREP='R5', PPLOC='R6',
                PPMAIN='R4')

        R1 ('HIT')
        R2 ('BARK')
        R3 ('JOHN')
        R4 ('MARY')
        R5 ('ON')
        R6 ('SWING')
        R7 ('IN')
        R8 ('PARK')
        R9 ('YESTERDAY')
        R10 ('DOG')

EXAMPLE OF ATTRIBUTE-TO-RELATION PROCESSING

"JOHN HIT MARY ON THE SWING IN
THE PARK YESTERDAY WHILE THE DOG BARKED."

FIGURE 9.

INPUT

    INDICATORS:
        VERBPHIND   13-33
            PAST   13,   PLUR   15
        VERBSW   49

    ATTRIBUTES:
        SUP   1

    NAMED RECORDS:
        RELLIST (@11='R1', @12='R2', @13='R3', @14='R4',
        '   @15='R5', @16='R6', @17='R7', LASTREC=17)
        R1 ('REL1', PP='D1', ACT='D2', PAST)
        R2 ('REL6', ACT='D2', PPOBJ='D8')
        R3 ('REL5A', PPMAIN='D8', PPLOC='D9', LOCPREP='D3')
        R4 ('REL13', MAINREL='R1', LOCPREP='D10', PPLOC='D6'
        R5 ('REL11', MAINREL='R1', TIME='D7')
        R6 ('REL12', MAINREL='R1', DEPREL='R7')
        R7 ('REL1', PP='D4', ACT='D5', PAST)
        D1 ('JOHN')
        D2 ('HIT')
        D3 ('ON')
        D4 ('DOG')
        D5 ('BARK')
        D6 ('PARK')
        D7 ('YESTERDAY')
        D8 ('MARY')
        D9 ('SWING')
        D10 ('IN')

OUTPUT
        D1 ('JOHN')
        D2 ('HIT', AGENT='D1', LOCATION='D10', GOAL='D8',
                CONCURR='D5', TIME='D7', PAST)
        D3 ('ON', LOCOBJ='D9')
        D4 ('DOG')
        D5 ('BARK', AGENT='D4', PAST)
        D6 ('PARK')
        D7 ('YESTERDAY')
        D8 ('MARY', LOCATION='D3')
        D9 ('SWING')
        D10 ('IN', LOCOBJ='D6')

        EXAMPLE OF RELATION-TO-ATTRIBUTE PROCESSING

            "JOHN HIT MARY ON THE SWING IN
        THE PARK YESTERDAY WHILE THE DOG BARKED."

                    FIGURE 10.

63

V. CONCLUSIONS

The objective of this research, as stated in the INTRODUCTION, has been met. First, a means was devised for representing, within the framework of NLP, the conceptual structure defined by Conceptual Dependency theory. Then two sets of rules, written in the rule language of NLP, were developed. The first set converts the attribute-value representation of information to dependency relation form, and the second set converts the dependency relation representation of information to attribute-value form.

Most of the information required by the conceptual structure is available in the existing structure of NLP, and that which is not currently available could be made available by making some minor modifications to NLP's present set of English processing rules.

It is recommended that, if further use is to be made in NLP of the conceptual dependency structure, consideration be given to modifying the processing rules of NLP so that they will produce the conceptual structure directly, rather than generating the present form of representation and then converting that to the conceptual structure.

LIST OF REFERENCES

1. Heidorn, George E., "Natural Language input to a simulation programming system," Technical Report NPS-55HD72101A, Naval Postgraduate School, Monterey, Calif., Oct. 1972.

2. Lamb, S.M., Outline of a Stratificational Grammar, Revised edition, Washington, D.C.: Georgetown University Press, 1966.

3. Schank, R. C., Goldman, N., Rieger, C. J., and Riesbeck, C. K.,"Primitive concepts underlying verbs of thought," AI Memo 164, Comp. Sci. Dept., Stanford Univ., Stanford, Calif., Feb. 1972.

4. Schank, R. C., "Identification of conceptualizations underlying natural language," Comp. Sci. Dept., Stanford Univ., Stanford, Calif., Feb. 1972.

5. Simmons, R. F., "Answering English questions by computer: a survey," COMM ACM 8, 1 (Jan. 1965), 53-70.

6. Simmons, R. F., "Natural language question-answering systems: 1969," COMM ACM 13, 1 (Jan. 1970), 15-30.

7. Simmons, R. F., and Slocum, J., "Generating English discourse from semantic networks," COMM ACM 15, 10 (Oct. 1972), 891-905.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Documentation Center   2
   Cameron Station
   Alexandria, Virginia 22314

2. Library, Code 0212   2
   Naval Postgraduate School
   Monterey, California 93940

3. Assistant Professor G. E. Heidorn, Code 55Hd   3
   Department of Operations Research and
       Administrative Sciences
   Naval Postgraduate School
   Monterey, California 93940

4. Visiting Professor D. K. Jefferson, Code 53Jf   1
   Department of Mathematics
   Naval Postgraduate School
   Monterey, California 93940

5. LT Bradley W. Hull   1
   NAVSEC
   Center Bldg.
   Prince George's Center
   Hyattsville, Maryland 20782

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3. REPORT TITLE

Conceptual Dependency Structures in the NLP
Natural Language Processor

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*
Master's Thesis; (December 1972)

5. AUTHOR(S) *(First name, middle initial, last name)*

Bradley Wayne Hull

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1972	68	7

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

There have been many systems developed for computer processing of natural languages such as English. One of these, known as NLP, is being developed at the Naval Postgraduate School. Another system, based on Conceptual Dependency theory, is being developed at Stanford University. The two systems, while having somewhat similar goals, use different internal representations of information.

The purpose of this thesis was to devise a means for representing the structures of Conceptual Dependency theory in NLP, and to develop methods for conversion of information between NLP's existing representation and that of Conceptual Dependency theory.

DD FORM 1473 (PAGE 1)
1 NOV 65
0101-807-6811

67

Unclassified
Security Classification
A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
NLP						
Natural Language Processor						
Conceptual Dependency Theory						